Vehicle Data Aggregator

Software Requirement Specification

(Version 1.0.0)

**Customer: Royal Enfield**

|  |  |  |  |
| --- | --- | --- | --- |
| SI. No | Prepared By | Review By | Version No |
| 1 | Jayanth | Hareesha/Aparna | 1.0.0 |
| 2 | Venu | Hareesha/Aparna | 1.0.0 |

Table of Contents

1. **INTRODUCTION…………………………………………………………………… 3**

1.1 Purpose……………………………………………………………………………..3

1.2 Scope……………………………………………………………………………….4

1.3 Definitions, Acronyms, and Abbreviations………………………………………...4

1. **OVERALL DESCRIPTION………………………………………………………....4**

2.1 Product Perspective………………………………………………………………...4

2.2 Product Functions…………………………………………………………………..5

2.2.1 Hardware Communication Layer (UART) ………………………………….....5

2.2.2 Communication Driver Layer…………………………………………………..5

2.2.3 Data I/O Interface ……………………………………………………………...5

2.2.4 Data Preserve Layer (Ring Buffer) …………………………………………….5

2.2.5 Signal Conversion Layer ………………………………………………………5

2.2.6 Common Data Layer …………………………………………………………..6

2.3 User Characteristics………………………………………………………………..6

1. **SPECIFIC REQUIREMENTS ……………………………………………………...6**

3.1 Functional Requirements ………………………………………………………….6

3.1.1 Message Reception ……………………………………………………………6

3.1.2 Data Processing and Aggregation …………………………………………….6

3.1.3 Data Transmission ………………………………………………………….....7

3.2 Non-Functional Requirements ……………………………………………………..7

3.2.1 Performance ……………………………………………………………………7

3.2.2 Reliability ………………………………………………………………………8

1. **INTERFACES ………………………………………………………………………...8**

4.1 External Interfaces………………………………………………………………….8

4.1.1 UART Interface ………………………………………………………………..8

4.2 Internal Interfaces …………………………………………………………………8

4.2.1 VCAN driver…………………………………………………………………...8

**5. CAN SIGNAL REQUIREMENTS…………………………………………………...8**

1. **Introduction**

The Software Requirements Specification (SRS) document for the CAN Data Aggregator System provides a comprehensive framework outlining the functional and non-functional requirements, intended functionalities, and the constraints of a system designed to aggregate and process data from multiple Electronic Control Units (ECUs) connected to a Controller Area Network (CAN) bus. The primary purpose of this system is to facilitate the collection of diverse data types transmitted by ECUs, such as sensor data, control commands, and status updates, and to consolidate this information into a centralized, coherent dataset. By doing so, the CAN Data Aggregator acts as a crucial intermediary, enabling the effective reception, organization, and subsequent transmission of vehicle data for further analysis, processing, and decision-making purposes. This introduction sets the stage for detailing the system's capabilities, the technological and operational environment in which it functions, and the specific user and system requirements necessary to achieve its objectives. Through its comprehensive aggregation and processing capabilities, the CAN Data Aggregator ensures that data collected from the UART is efficiently organized and transmitted to cloud platforms.

* 1. **Purpose**

This document outlines the Software Requirements Specification (SRS) for the Vehicle Data Aggregator, a software module responsible for collecting, processing, and transmitting CAN bus data from Electronic Control Units (ECUs) to a cloud platform.

1. **Centralized Data Management:** It provides a unified platform for aggregating data from disparate ECUs, enabling a comprehensive view of vehicle status, performance, and diagnostics information. This centralized approach simplifies data analysis and decision-making processes by collating all necessary information in one place.
2. **Enhanced Vehicle Diagnostics and Monitoring:** By gathering and processing data from multiple sources within the vehicle, the Vehicle Data Aggregator facilitates more accurate and detailed diagnostics, allowing for the early detection of potential issues and proactive maintenance, thereby improving vehicle reliability and safety.
3. **Data Accessibility for Third-party Applications:** The system ensures that aggregated vehicle data is structured and made accessible to third-party applications and services.
4. **Optimization of Vehicle Performance:** Through the analysis of collected data, the Vehicle Data Aggregator can identify patterns and insights that lead to recommendations for optimizing vehicle performance and efficiency, such as fuel consumption, battery life in electric vehicles, and dynamic vehicle adjustments.
   1. **Scope**

This document defines the functionalities, interfaces, and performance requirements of the Vehicle Data Aggregator. It excludes the functionalities of the ECUs, Body Control Module (BCM), and the cloud platform.

**1.3 Definitions, Acronyms, and Abbreviations**

* + **ECU** - Electronic Control Unit
  + **CAN** - Controller Area Network
  + **SOM** - System on Module
  + **UART** - Universal Asynchronous Receiver/Transmitter
  + **VCAN** - Virtual CAN
  + **MTU** - Maximum Transmission Unit

1. **Overall Description**
   1. **Product Perspective**

The Vehicle Data Aggregator resides within the System on Module (SoM) of a vehicle. It interacts with the Body Control Module (BCM) through UART and utilizes Virtual CAN (VCAN) to interpret CAN messages.

1. **System Integration:**

Describe how the Vehicle Data Aggregator fits into the larger vehicle management system, including its role in connecting with Electronic Control Units (ECUs) through the CAN bus and its interaction with external cloud services for data analytics and storage.

1. **System Interfaces:**

Outline the interfaces between the Vehicle Data Aggregator and other system components, including hardware interfaces and software interfaces.

**2.2 Product Functions**

**2.2.1 Hardware Communication Layer (UART)**

The Vehicle Data Aggregator shall establish a physical connection with the BCM using the UART communication protocol at a pre-defined baud rate and configuration.

* + 1. **Communication Driver Layer**
* The Vehicle Data Aggregator shall receive electrical signals from the UART layer and convert them into a digital format for further processing.
* The communication driver shall handle potential errors during data reception, such as framing errors or parity errors.
  + 1. **Data I/O Interface**
* The Vehicle Data Aggregator shall provide a single interface for upper layers to access data from CAN messages.
* The Vehicle Data Aggregator shall abstract CAN data using the following criteria:
  + - * CAN ID (identifier for the specific message)
      * Data Length (number of bytes in the message)
      * Data Payload (actual data content of the message) - Define data type (e.g., integer, float) for each byte in the payload based on message specification.
      * Periodicity (frequency at which the message is transmitted)
    1. **Data Preserve Layer (Ring Buffer)**
* The Vehicle Data Aggregator shall utilize a ring buffer to temporarily store incoming CAN data.
* The ring buffer size shall be configurable to balance memory usage and data loss prevention.
* The Vehicle Data Aggregator shall define a mechanism to handle buffer overflow scenarios (e.g., dropping oldest data).
  + 1. **Signal Conversion Layer**
* The Vehicle Data Aggregator shall convert raw CAN frames into engineering signals suitable for further processing.
* The Vehicle Data Aggregator shall offer configurable options for storing raw CAN frames with different periodicities:
* Fastest: 10 milliseconds
* Moderate: 50 milliseconds
* Slow: 500 milliseconds
* The Vehicle Data Aggregator shall provide an option to enable user-defined algorithms for raw CAN frame processing or filtering (configurable).
* The Vehicle Data Aggregator shall store a configurable amount of the latest raw CAN data (e.g., 15 minutes) based on SOM memory availability.
  + 1. **Common Data Layer**
* The Vehicle Data Aggregator shall provide a common data layer as the source for external layers.
* The common data layer shall offer interfaces for various data delivery strategies, including:
* Intent-based broadcasting
* AIDL (Android Interface Definition Language)
  1. **User Characteristics**

The Vehicle Data Aggregator is not intended for direct user interaction. It operates in the background to facilitate communication between ECUs and the cloud platform.

**3. Specific Requirements**

* 1. **Functional Requirements**

**3.1.1 Message Reception**

* + - The Vehicle Data Aggregator shall receive CAN messages from the BCM via UART at a pre-defined baud rate and configuration.
    - The Vehicle Data Aggregator shall be able to decode and interpret CAN messages with various formats and identifiers as defined by the CAN bus specification

**3.1.2 Data Processing and Aggregation**

* + - The Vehicle Data Aggregator shall parse the received CAN messages and extract relevant data based on pre-defined configurations.
    - The Vehicle Data Aggregator shall combine data from different ECUs into a unified data structure based on user-defined criteria (e.g., message type, timestamp).
    - The Vehicle Data Aggregator shall allow configuration of data organization based on desired structure, type, and size (e.g., packing multiple data points into a single message for efficiency).
    1. **Data Transmission**
* The Vehicle Data Aggregator shall establish a secure connection with the cloud platform using a configurable protocol (e.g., MQTT, secure socket layer).
* The Vehicle Data Aggregator shall be configurable for data transmission parameters like packet size and frequency based on data type, priority, and network bandwidth limitations.
* The Vehicle Data Aggregator shall adhere to the MTU limitations of the network to avoid data fragmentation.
* The Vehicle Data Aggregator shall distinguish and handle data from different ECUs based on CAN ID or other identifiers for optimized transmission.
* The Vehicle Data Aggregator shall categorize messages into different event types (low priority, high priority, high frequency) for prioritized transmission.
  + - * **Low priority events**: These messages contain non-critical or non-urgent information and may be transmitted at lower frequencies (e.g., vehicle status updates).
      * **Moderate priority events**: These messages are critical and time-sensitive, requiring immediate attention or action.
      * **High priority events**: These messages contain real-time or near real-time data that needs to be updated and processed rapidly.
  1. **Non-Functional Requirements**
     1. **Performance**
* The Vehicle Data Aggregator shall have minimal latency in processing and transmitting CAN data, meeting a defined threshold for different event types (e.g., low latency for high priority events).
* The Vehicle Data Aggregator shall be able to handle high volumes of CAN messages without compromising performance (specify message throughput per second).
  + 1. **Reliability**
* The Vehicle Data Aggregator shall operate reliably with minimal downtime. Define uptime target (e.g., 99.9%).
* The Vehicle Data Aggregator shall handle errors gracefully and implement mechanisms for recovery (e.g., retry logic for failed transmissions).

**4. Interfaces**

**4.1 External Interfaces**

* + 1. **UART Interface**
    - The UART interface provides communication with the BCM. It shall be configurable for baud rate, parity, and stop bits according to the system specifications.
    - The Vehicle Data Aggregator shall handle potential errors during UART communication.

**4.1.2 Cloud Platform Interface**

* + - The cloud platform interface provides communication with the cloud for data transmission. The specific protocol (e.g., MQTT, secure socket layer) and configuration options shall be defined based on the chosen cloud platform.
  1. **Internal Interfaces**
     1. **VCAN Driver**
* The VCAN driver interface enables the Vehicle Data Aggregator to interpret CAN messages received through UART via the VCAN software module.

1. **CAN Signal Requirements**

The following table specifies the properties of each CAN signal that the Vehicle Data Aggregator system will process. The CAN ID represents the unique identifier for each message on the bus, which is tied to a specific ECU. The CAN ID Name provides a human-readable name for the ECU or module the ID belongs to. The Signal Name is the specific data point being monitored, such as the ABS hardware version. Minimum and Maximum Values define the range of possible values the signal can take. The Factor and Offset are used to convert the raw data into human-readable form and correct for any sensor-specific deviations.

Table 1: **BATTERY\_LIMITS (0x12F)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CAN ID | Signal Name | Offset | Factor | Minimum | Maximum | Unit |
| 0x12F | Charge\_Current\_Limit | 0 | 0.01 | 0 | 600 | Amp |
| 0x12F | Discharge\_Current\_Limit | -400 | 0.01 | -400 | 0 | Amp |

Table 2: **BATTERY\_TEMPERATURE (0x172)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CAN ID | Signal Name | Offset | Factor | Minimum | Maximum | Unit |
| 0x172 | Battery\_Temperature\_1 | -50 | 0.5 | -50 | 77.5 | degC |
| 0x172 | Battery\_Temperature\_2 | -50 | 0.5 | -50 | 77.5 | degC |
| 0x172 | Battery\_Temperature\_3 | -50 | 0.5 | -50 | 77.5 | degC |
| 0x172 | Battery\_Temperature\_4 | -50 | 0.5 | -50 | 77.5 | degC |
| 0x172 | Battery\_Temperature\_5 | -50 | 0.5 | -50 | 77.5 | degC |
| 0x172 | Battery\_Temperature\_6 | -50 | 0.5 | -50 | 77.5 | degC |
| 0x172 | Battery\_Temperature\_Min | -50 | 0.5 | -50 | 77.5 | degC |
| 0x172 | Battery\_Temperature\_Max | -50 | 0.5 | -50 | 77.5 | degC |

* **CAN ID:** The hexadecimal identifier for the ECU message.
* **CAN ID Name:** A descriptive name for the source of the CAN message.
* **Signal Name:** The specific signal or data point contained within the CAN message.
* **Minimum Value:** The lowest possible value that can be read from the signal.
* **Maximum Value:** The highest possible value that can be read from the signal.
* **Factor:** The scaling factor used for the signal. The raw data from the CAN message will be multiplied by this factor.
* **Offset:** The offset value to be added or subtracted from the scaled signal to obtain the correct value